가상화 지원 프로그래머블 플랫폼을 위한 제어프레임워크

정상진*°, 신명기*, 김형준*, 윤찬현°
*한국전자통신연구원 표준연구센터
°KAIST 전기및전자공학과/정보통신공학과
e-mail : sjjeong@etri.re.kr

A Control Framework for Virtualization Support Programmable Platform

Sangjin Jeong*°, Myung-Ki Shin*, Hyoung-Jun Kim*, Chan-Hyun Youn°
*Standards Research Center, ETRI
°Dept. of Electrical Engineering/Dept. of Information & Communications Engineering, KAIST

요 약

본 논문에서는 미래인터넷의 핵심 자원 중 하나인, 네트워크 가상화 지원 프로그래머블 플랫폼 또는 라우터의 제어프레임워크 구조를 제안한다. 본 논문에서 제안된 제어프레임워크는 네트워크 프로세서 기반의 프로그래머블 플랫폼 상에서 사용자의 요청에 따라 실시간으로 네트워크 프로세서에 동적으로 슬리버를 생성하고, 생성된 슬리버를 슬라이스로 확장할 수 있는 기능을 제공할 수 있다. 이를 통해 생성된 슬라이스 위에서 사용자의 요구사항 별 다양한 미래인터넷 서비스를 동시에 서로 간섭 없이 실행할 수 있다.

1. 서론

미래인터넷의 핵심 기능 중 하나는 하나의 물리적인 네트워크 위에서 여러 개의 가상 네트워크를 구축할 수 있는 네트워크 가상화 기술이다. 네트워크 가상화는 현 네트워크 구조의 경직성을 해결하기 위한 방안으로서, 네트워크 링크와 노드를 포함한 네트워크 내 모든 자원을 가상화 하여 하나의 물리적인 인프라상에서 요구사항이 서로 다른 응용/서비스/이용자별로 가상 네트워크들이 상호 독립적으로 공존할 수 있게 하는 기술이다. 네트워크 가상화를 통하여, 이용자는 단일 네트워크 인프라상에서 서로 다른 응용/서비스/이용자들간 독립적으로 가상 네트워크를 만들 수 있다. 이러한 네트워크 가상화를 지원하는 미래인터넷 상의 자원들(라우터, PC, 클러스터 노드 등)을 제어하기 위해서 다양한 제어프레임워크들이 개발되고 있다. 하지만 이들 다양한 제어 프레임워크는 각각의 대상 자원을 한정하고 있다. 예를 들어 ProtoGENI [1]는 Emulab 노드를 대상으로 하고 있으며, PlanetLab[2]은 PC 를 대상으로 하고 있다. 더구나, 아직까지 프로그래머블 라우터와 같은 프로그래머블 플랫폼, 특히 네트워크 프로세서 기반의 고속 프로그래머블 플랫폼을 관리할 수 있는 제어프레임워크는 개발되어 있지 않다.

본 논문에서는 미래인터넷의 핵심 자원 중 하나인, 네트워크 가상화 지원 프로그래머블 플랫폼 또는 라우터의 제어프레임워크 구조를 제안한다. 본 논문에서 제안된 제어프레임워크는 네트워크 프로세서 기반의 프로그래머블 플랫폼 상에서 사용자의 요청에 따라 실시간으로 네트워크 프로세서에 동적으로 슬리버를 생성하고, 생성된 슬리버를 슬라이스로 확장할 수 있는 기능을 제공할 수 있다. 이를 통해 생성된 슬라이스 위에서 사용자의 요구사항 별 다양한 미래인터넷 서비스를 동시에 서로 간섭 없이 실행할 수 있다.

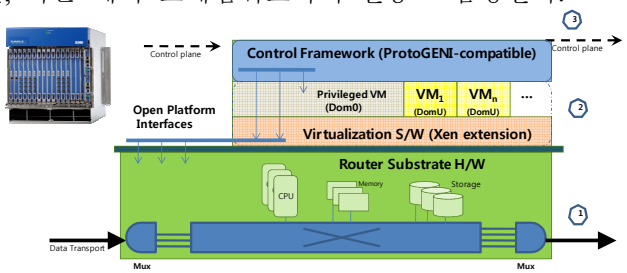
2. 네트워크 가상화 지원을 위한 프로그래머블 플랫폼 구조

본 장에서는 네트워크 가상화 지원 프로그래머블 플랫폼의 구조에 대하여 설명한다. 네트워크 가상화 지원을 위해서는 다음의 설계원칙을 적용해야 한다[3].

- 유연성 (Flexibility): 다양한 네트워킹 기술의 적용이 쉬워야 함
- 관리 용이성 (Manageability): 네트워크 관리 업무를 모듈화 해서 관리의 용이성을 증진시킴
- 확장성 (Scalability): 여러 개의 가상 네트워크가 동시에 존재할 수 있어야 함
- 독립성 (Isolation): 각 가상 네트워크는 서로 독립적으로 구성되고 운용될 수 있어야 함
- 안정성 (Stability and Convergence): 한 가상 네트워크의 오동작이 다른 가상 네트워크에 영향을 주지 않아야 함
- 프로그램 용이성 (Programmability): 유연성과 관리 용이성을 위해 네트워크 구성 요소들을 프로그래

밍 할 수 있어야 함
- 이질성 (Heterogeneity): 서로 다른 구조의 가상 네트워크가 공존할 수 있어야 함
- 구 호환 (Legacy Support): 이전의 네트워크와 연동될 수 있어야 함

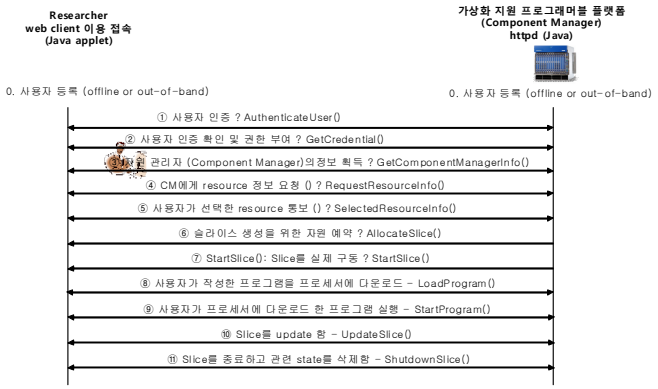
[그림 1]에 도시된 것처럼, 프로그래머블 플랫폼은 플랫폼 하드웨어 계층, 가상화 지원을 위한 가상화 S/W 계층과 플랫폼의 제어를 위한 제어 프레임워크로 구성된다. 하드웨어 계층은 플랫폼 또는 라우터의 기본 구성요소인 CPU, 메모리, 네트워크 라인카드 등을 포함한다. 가상화 S/W 계층은 가상화를 지원하지 않는 플랫폼에서 가상화 기능 지원을 위해 탑재하는 가상화 지원 S/W 이다. 가상화 S/W 는 기존 서버 가상화 등의 기술에서 사용되는 가상화 S/W 를 확장하여 적용할 수 있다. 마지막으로 제어프레임워크는 프로그래머블 플랫폼과 가상화 S.W 전체를 제어하는 기능을 담당하며, 다른 프로그래머블 플랫폼과 연합하여 네트워크 슬라이스를 생성할 수 있도록 해준다. 또한, 다른 제어 프레임워크와의 연동도 담당한다.



[그림 1] 네트워크 가상화 지원 프로그래머블 플랫폼 구조

3. 네트워크 가상화 지원 프로그래머블 플랫폼을 위한 제어 프레임워크 구조

본 장에서는 네트워크 가상화를 지원하는 프로그래머블 플랫폼의 제어를 위한 제어 프레임워크의 구조에 대해 기술한다. [그림 2]는 본 논문에서 제안한 가상화 지원 프로그래머블 플랫폼의 제어프레임워크를 나타낸 것이다. 사용자는 별도의 시그널링 방법을 통해 사용자 계정을 등록한다. 그 후, 사용자는 웹 기반으로 가상화 지원 프로그래머블 플랫폼에 접속하여 인증 및 사용 권한을 확인한다. 사용자의 인증은 AuthenticateUser() API 를 이용하여 수행하고, 사용 권한 확인 및 부여는 GetCredential() API 를 이용한다. 사용자 인증이 완료되면, 플랫폼의 컴포넌트 관리자에게 플랫폼 내의 자원 정보를 요청하고, 자원의 정보를 자원규격 (RSpec)으로 수신한다. 이를 위해 GetComponentManagerInfo(), RequestResourceInfo() API 를 이용한다. 그 후, 수신된 RSpec 을 이용하여 자원을 선택하여 플랫폼에 통보한다 (SelectedResourceInfo()). RSpec 을 이용하여 사용자가 선택한 자원 정보를 전달받은 플랫폼은 가상화된 플랫폼 환경인 슬라이스를 생성한다 (AllocateSlice(), StartSlice() API). 슬라이스의 생성이 완료되면, 사용자 프로그램을 슬라이스에 다운로드 하여 프로그램을 실행한다 (LoadProgram(), StartProgram() API). 생성된 슬라이스의 변경이 필요하면 UpdateSlice() API 를 이용해서 슬라이스 정보를 수정하게 되고, 수행을 종료하기 위해서는 ShutdownSlice() API 를 이용하여 수행을 종료한다.



[그림 2] 네트워크 가상화 지원 제어프레임워크 절차

제어프레임워크에서 플랫폼의 자원 상태를 사용자에게 통보(통보 RSpec)하고, 사용자가 선택한 자원을 제어프레임워크로 전달(전달 RSpec)하기 위해서는 RSpec 을 사용한다. RSpec 은 요청 RSpec 과 통보 RSpec 으로 구분된다. RSpec 은 XML 형식으로 표현된다. RSpec 에서 플랫폼 노드의 표현 방식은 아래와 같다

Node description (노드의 주소 URL, ID, 이름, 인터페이스 수, CPU, 메모리, 저장공간)
node_component_manager_URL="http://firstcm1.etri.re.kr"
node_component_name=node_CM_name
node_component_uuid=UUID
node_type=router/PC(?)
node_interface_type=ethernet-10/ethernet-100/ethernet-1000/ethernet-10000
node_interface_slots=number_of_interfaces
node_storage_capacity=KB/MB/TB
node_processing_units=MHz/GHz
node_memory=KB/MB/TB

Link description (링크의 주소, ID, 이름, 링크 종류, 대역폭, 지연, 손실률)
link_component_manager_URL="http://firstcm2.etri.re.kr"
link_component_name=link_CM_name
link_component_uuid=UUID
link_type=ethernet
link_packet_loss=number_of_interfaces
link_bandwidth=Kbps/Mbps/Gbps
link_latency=nsec/usec/msec

4. 결론

본 논문에서 제안한 제어프레임워크는 네트워크 프로세서 기반의 프로그래머블 플랫폼 상에서 사용자의 요청에 따라 실시간으로 네트워크 프로세서에 동적으로 슬리버를 생성하고, 생성된 슬리버를 슬라이스로 확장할 수 있는 기능을 제공할 수 있다.

참고문헌

[1] http://www.protogeni.net
[2] http://www.planet-lab.org
[3] M. Chowdhury, R. Boutaba, Network Virtualization: State of the Art and Research Challenges, IEEE Comm. Mag. Vol.47, No.7, July 2009.